Transcrição

Depois de completarmos os exercícios do notebook proposto, faremos uma **comparação** para fazermos eventuais **correções**.

No passo anterior acessível por este **link**, encontraremos o download do arquivo `Análise_Descritiva_Resposta.ipynb` com todas as respostas con

Na primeira parte, importamos a biblioteca `pandas` como `pd`, `numpy` como `np` e `seaborn` como `sns`.

```
import pandas as pd
import numpy as np
import seaborn as sns
```

Depois, importamos o dataset `dados` como um `DataFrame` por meio da leitura do arquivo `dados.csv` com a função `read_csv()`.

```
dados = pd.read_csv('dados.csv')
```

Visualizamos os cinco primeiros registros do `dados` com `head()`.

```
dados.head()
```

	UF	Sexo	Idade	Cor	Anos de Estudo	Renda	Altura
0	11	0	23	8	12	800	1.603808
1	11	1	23	2	12	1150	1.739790
2	11	1	35	8	15	880	1.760444
3	11	0	46	2	6	3500	1.783158
4	11	1	47	8	9	150	1.690631

Em seguida, criamos uma Distribuição de Frequências para a variável `Renda`, como as já conhecidas classes "A", "B", "C", "D" e "E". Aqui, criamo mínimos que cada divisão contém:

```
classes = [
    dados.Renda.min(),
    2 * 788,
    5 * 788,
    15 * 788,
    25 * 788,
    dados.Renda.max()
]
classes
```

Como retorno, teremos uma lista com os valores de renda que definem os limites de cada classe, do menor para o maior.

```
[0, 1576, 3940, 11820, 19700, 200000]
```

Em seguida, definimos os labels da classificação em `E`, `D`, `C`, `B` e `A` de acordo com o retorno anterior.

```
labels = ['E', 'D', 'C', 'B', 'A']
```

Construímos a coluna de frequências com a variável `frequencia` como fizemos em nossas aulas. Utilizamos o `value_counts()` com `cut()` para

```
frequencia = pd.value_counts(
    pd.cut(x = dados.Renda,
           bins = classes,
           labels = labels,
           include_lowest = True)
)
frequencia
```

```
E    49755
D    18602
C     7241
B      822
A      420
Name: Renda, dtype: int64
```

Aplicamos a mesma metodologia para construirmos a coluna de percentuais com a variável `percentual` normalizada e multiplicada por `100`.

```
percentual = pd.value_counts(
    pd.cut(x = dados.Renda,
           bins = classes,
           labels = labels,
           include_lowest = True),
    normalize = True
) * 100
percentual
```

```
E    64.751432
D    24.208745
C     9.423477
B     1.069755
A     0.546590
Name: Renda, dtype: float64
```

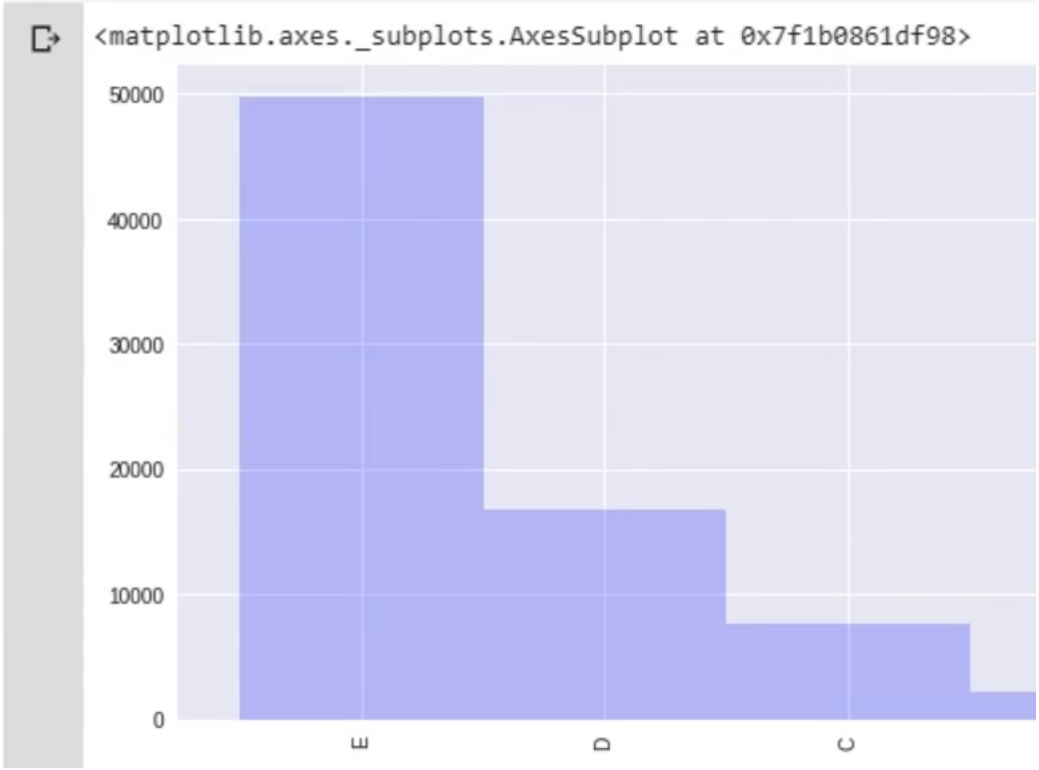
Com isso, criamos a variável `dist_freq-renda` , juntamos as duas colunas e ordenamos as linhas alfabeticamente de acordo com os labels de cada c

```
dist_freq_renda = pd.DataFrame(
    {'Frequência': frequencia, 'Porcentagem (%)': percentual}
)
dist_freq_renda.sort_index(ascending = False)
```

	Frequência	Porcentagem (%)
A	420	0.546590
B	822	1.069755
C	7241	9.423477
D	18602	24.208745
E	49755	64.751432

Gerada a Tabela de Frequências, construímos uma representação gráfica para visualizarmos suas informações com o método `.plot.bar()` .

```
dist_freq_renda['Frequência'].plot.bar(width = 1, color = 'blue', alpha = 0.2, figsize=(14, 6))
```



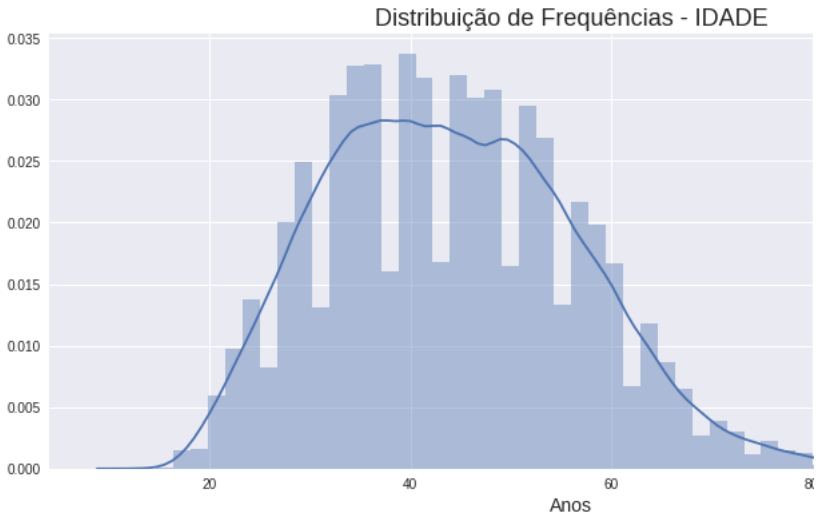
Não se trata de um Histograma, pois este é feito com variáveis quantitativas, e como criamos uma variável qualitativa, utilizamos um gráfico de barr

Adiante, criamos um Histograma com `distplot()` para as variáveis quantitativas de nosso dataset:.

- `Idade` :

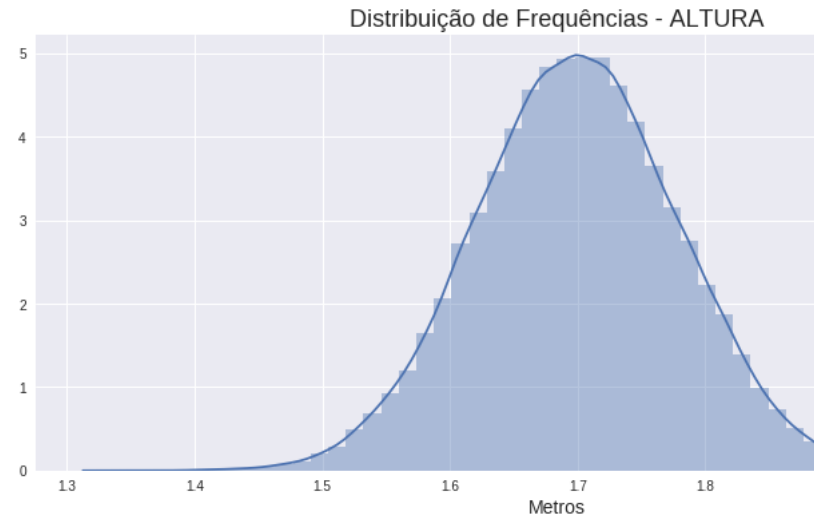
```
ax = sns.distplot(dados['Idade'])
ax.figure.set_size_inches(14, 6)
ax.set_title('Distribuição de Frequências - IDADE', fontsize=18)
```

```
ax.set_xlabel('Anos', fontsize=14)
ax
```



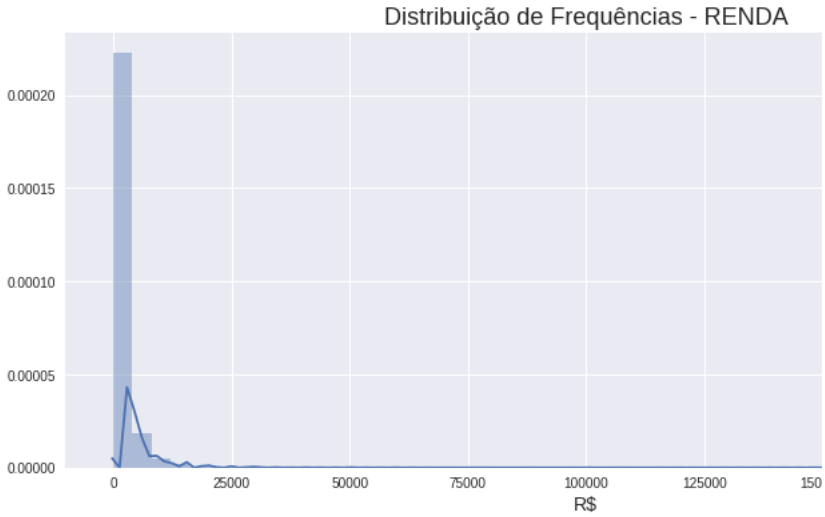
- Altura :

```
ax = sns.distplot(dados['Altura'])
ax.figure.set_size_inches(14, 6)
ax.set_title('Distribuição de Frequências - ALTURA', fontsize=18)
ax.set_xlabel('Metros', fontsize=14)
ax
```



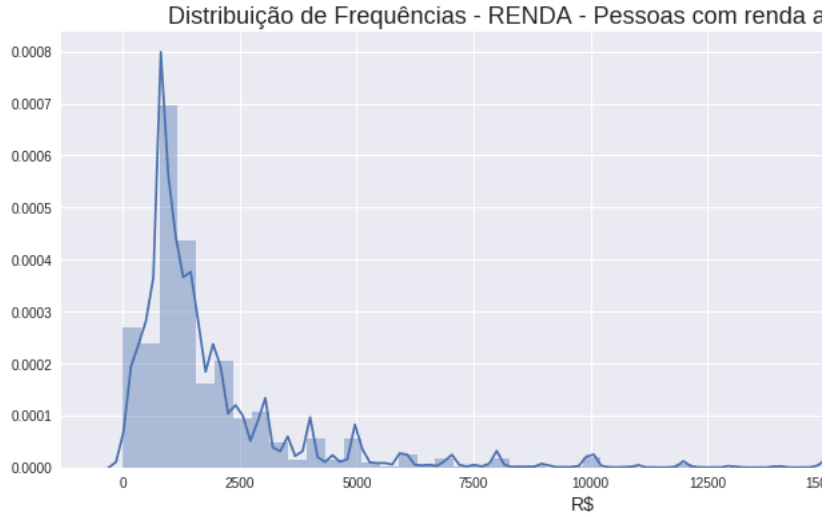
- Renda :

```
ax = sns.distplot(dados['Renda'])
ax.figure.set_size_inches(14, 6)
ax.set_title('Distribuição de Frequências - RENDA', fontsize=18)
ax.set_xlabel('R$', fontsize=14)
ax
```



Ao final, temos um espaço para escrevermos nossas conclusões a partir da análise dessas informações.

Para visualizarmos melhor os valores de `Renda` e sua simetria brusca, fizemos uma seleção com `query()` com somente pessoas cuja renda é até R$



Em seguida, construímos uma Tabela de Frequências e de percentuais a partir de dicionários usados para nomearmos os registros de acordo com o da

```
sexo = {
    0: 'Masculino',
    1: 'Feminino'
}
cor = {
    0: 'Indígena',
    2: 'Branca',
    4: 'Preta',
    6: 'Amarela',
    8: 'Parda',
    9: 'Sem declaração'
}
anos_de_estudo = {
    1: 'Sem instrução e menos de 1 ano',
    2: '1 ano',
    3: '2 anos',
```

```
    4: '3 anos',
    5: '4 anos',
    6: '5 anos',
    7: '6 anos',
    8: '7 anos',
    9: '8 anos',
    10: '9 anos',
    11: '10 anos',
    12: '11 anos',
    13: '12 anos',
    14: '13 anos',
    15: '14 anos',
    16: '15 anos ou mais',
    17: 'Não determinados'
}
uf = {
    11: 'Rondônia',
    12: 'Acre',
    13: 'Amazonas',
    14: 'Roraima',
    15: 'Pará',
    16: 'Amapá',
    17: 'Tocantins',
    21: 'Maranhão',
    22: 'Piauí',
    23: 'Ceará',
    24: 'Rio Grande do Norte',
    25: 'Paraíba',
    26: 'Pernambuco',
    27: 'Alagoas',
    28: 'Sergipe',
    29: 'Bahia',
    31: 'Minas Gerais',
    32: 'Espírito Santo',
    33: 'Rio de Janeiro',
    35: 'São Paulo',
    41: 'Paraná',
    42: 'Santa Catarina',
    43: 'Rio Grande do Sul',
    50: 'Mato Grosso do Sul',
    51: 'Mato Grosso',
    52: 'Goiás',
    53: 'Distrito Federal'
}
```

Em seguida, utilizamos os dicionários e geramos uma tabela `frequencia` com o cruzamento das duas variáveis por meio do método `crosstab()` .

```
frequencia = pd.crosstab(dados.Sexo,
                         dados.Cor
                        )
frequencia.rename(index = sexo, inplace = True)
frequencia.rename(columns = cor, inplace = True)
frequencia
```

Cor	Indígena	Branca	Preta	Amarela	Parda
Sexo					
Masculino	256	22194	5502	235	25063
Feminino	101	9621	2889	117	10862

Se não tivéssemos utilizado as duas informações dentro das linhas de `.rename()`, teríamos colunas com valores numéricos apenas.

A mesma técnica foi usada para a construção da tabela de `percentual`, com a multiplicação por `100`.

```
percentual = pd.crosstab(dados.Sexo,
                         dados.Cor,
                         normalize = True
                        ) * 100
percentual.rename(index = sexo, inplace = True)
percentual.rename(columns = cor, inplace = True)
percentual
```

Cor	Indígena	Branca	Preta	Amarela	Parda
Sexo					
Masculino	0.333160	28.883394	7.160333	0.305830	32.617126
Feminino	0.131442	12.520822	3.759761	0.152264	14.135867

Com isso, já puderemos fazer algumas análises.

Em seguida, fizemos uma análise descritiva da variável `Renda` e obtivemos sua média com `mean()`, a mediana com `median()` e moda com `mode`

Os resultados foram `2000.3831988547631`, `1200.0` e `788` respectivamente.

No passo seguinte, obtivemos o desvio médio absoluto com `mad()`, a variância com `var()` e o desvio padrão com `std()`.

```
dados.Renda.mad()
```

```
dados.Renda.var()
```

```
dados.Renda.std()
```

Também obtivemos a média, mediana e valor máximo da variável `Renda` de acordo com o `Sexo` e a `Cor` por meio dos métodos `crosstab()` nov do Pandas e recebe as três funções.

```
renda_estatisticas_por_sexo_e_cor = pd.crosstab(dados.Cor,
                                                dados.Sexo,
                                                values = dados.Renda,
                                                aggfunc = {'mean', 'median', 'max'})
renda_estatisticas_por_sexo_e_cor.rename(index = cor, inplace = True)
renda_estatisticas_por_sexo_e_cor.rename(columns = sexo, inplace = True)
renda_estatisticas_por_sexo_e_cor
```

	max		mean		median	
Sexo	Masculino	Feminino	Masculino	Feminino	Masculino	Feminino
Cor						
Indígena	10000	120000	1081.710938	2464.386139	797.5	788.0
Branca	200000	100000	2925.744435	2109.866750	1700.0	1200.0
Preta	50000	23000	1603.861687	1134.596400	1200.0	800.0
Amarela	50000	20000	4758.251064	3027.341880	2800.0	1500.0
Parda	100000	30000	1659.577425	1176.758516	1200.0	800.0

Com esta tabela, pudemos tirar algumas conclusões.

Em seguida, fizemos basicamente o mesmo procedimento para o cálculo das Estatísticas de Dispersão. Calculamos as medidas de desvio médio abso

```
renda_dispersao_por_sexo_e_cor = pd.crosstab(dados.Cor,
                                             dados.Sexo,
                                             aggfunc = {'mad', 'var', 'std'},
                                             values = dados.Renda).round(2)
renda_dispersao_por_sexo_e_cor.rename(index = cor, inplace = True)
renda_dispersao_por_sexo_e_cor.rename(columns = sexo, inplace = True)
renda_dispersao_por_sexo_e_cor
```

	mad		std		var	
Sexo	Masculino	Feminino	Masculino	Feminino	Masculino	Feminino
Cor						
Indígena	798.91	3007.89	1204.09	11957.50	1449841.13	1.429818e+08
Branca	2261.01	1670.97	4750.79	3251.01	22570023.41	1.056909e+07
Preta	975.60	705.45	1936.31	1349.80	3749293.59	1.821960e+06
Amarela	3709.60	2549.15	5740.82	3731.17	32957069.62	1.392166e+07
Parda	1125.83	811.58	2312.09	1596.23	5345747.15	2.547960e+06

Depois disso, partimos para a construção do `boxplot()` da variável `Renda` segundo `Sexo` e `Cor` .

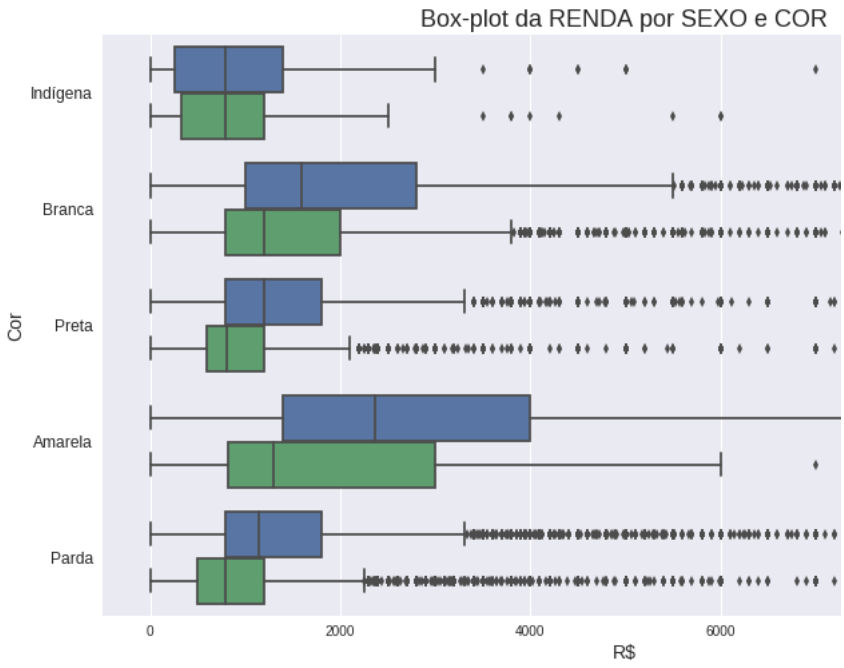
Nesta parte do notebook, encontramos algumas dicas com o uso do parâmetro `hue` para inserirmos uma outra nova variável.

Passamos o `x` sendo a `'Renda'`, o `y` como a `Cor` e a diferenciação de acordo com o `Sexo`. Por fim, configuraremos a visualização do gráfico c

O método `get_legend_handles_labels()` nos dá duas saídas, como uma tupla. O `handles` com `_` nos passa a legenda da divisão entre `'Mascul`

Fizemos uma seleção de rendas até R$10.000,00 para podermos obter resultados de mais fácil leitura.

```
ax = sns.boxplot(x = 'Renda', y = 'Cor', hue = 'Sexo', data=dados.query('Renda < 10000'), orient='h')

ax.figure.set_size_inches(14, 8)    # Personalizando o tamanho da figura

ax.set_title('Box-plot da RENDA por SEXO e COR', fontsize=18)    # Configurando o título do gráfico

ax.set_xlabel('R$', fontsize=14)    # Configurando o label do eixo X

ax.set_ylabel('Cor', fontsize=14)    # Configurando o label do eixo Y
ax.set_yticklabels(['Indígena', 'Branca', 'Preta', 'Amarela', 'Parda', 'Sem declaração'], fontsize=12)    #

# Configurações da legenda do gráfico (Sexo)
handles, _ = ax.get_legend_handles_labels()
ax.legend(handles, ['Masculino', 'Feminino'], fontsize=12)

ax
```



Mais conteúdos sobre Seaborn podem ser encontrados aqui na Plataforma Alura.

Com isso, poderemos ver claramente enormes diferenças de rendas tanto entre pessoas pertencentes à uma cor ou outra quanto entre homens e mulhe

O desafio seguinte propôs a obtenção do percentual exato de pessoas que ganham apenas um salário mínimo ou menos. Primeiro, importamos `stat`
a `Renda` de `dados`, o valor de corte `788` do salário mínimo e o `kind` igual a `'weak'`.

Não utilizamos os percentis, pois teríamos que continuar dividindo em mais partes até chegarmos em um valor exato.

O método `percentileofscore()` nos permite pegar este valor rapidamente.

```
from scipy import stats

percentual = stats.percentileofscore(dados.Renda, 788, kind = 'weak')
print("{0:.2f}%".format(percentual))
```

O resultado será `28.87%` das pessoas que ganham igual ou menos do que um salário mínimo de R$788,00 por mês.

Essa mesma parte possui [este link de acesso à documentação] do método. Veremos alguns usos do padrão `kind` na página.

No primeiro exemplo, há uma lista com `[1, 2, 3, 3, 4]` dentro de `percentileofscore()` seguido de `3`, que seria o valor de corte por exemplo

Quando usamos o `kind` como `'strict'` da mesma lista e corte, o resultado será 40%, pois informa que somente o que está abaixo do valor de cor

O `'weak'` que usamos já inclui o valor, pois queremos uma renda menor ou igual a um salário mínimo. No exemplo, o resultado é 80%.

Já o `'mean'` identifica a mediana. É interessante começarmos a lidar com a biblioteca `scipy`, pois é bastante útil.

Mais adiante, calcularemos o valor máximo que 99% das pessoas ganham em nosso dataset. Utilizaremos a função `quantile()` recebendo `.99` qu

```
valor = dados.Renda.quantile(.99)
print("R$ {0:.2f}".format(valor))
```

O resultado será `R$15000.00`.

Em seguida, obtivemos a média, mediana, valor máximo e desvio padrão de `Renda` de acordo com os `Anos de Estudo` e `Sexo`, utilizando o `cros`

```
renda_estatisticas_por_sexo_e_estudo = pd.crosstab(dados['Anos de Estudo'],
                                                   dados.Sexo,
                                                   aggfunc = {'mean', 'median', 'max', 'std'},
                                                   values = dados.Renda).round(2)
renda_estatisticas_por_sexo_e_estudo.rename(index = anos_de_estudo, inplace = True)
renda_estatisticas_por_sexo_e_estudo.rename(columns = sexo, inplace = True)
renda_estatisticas_por_sexo_e_estudo
```

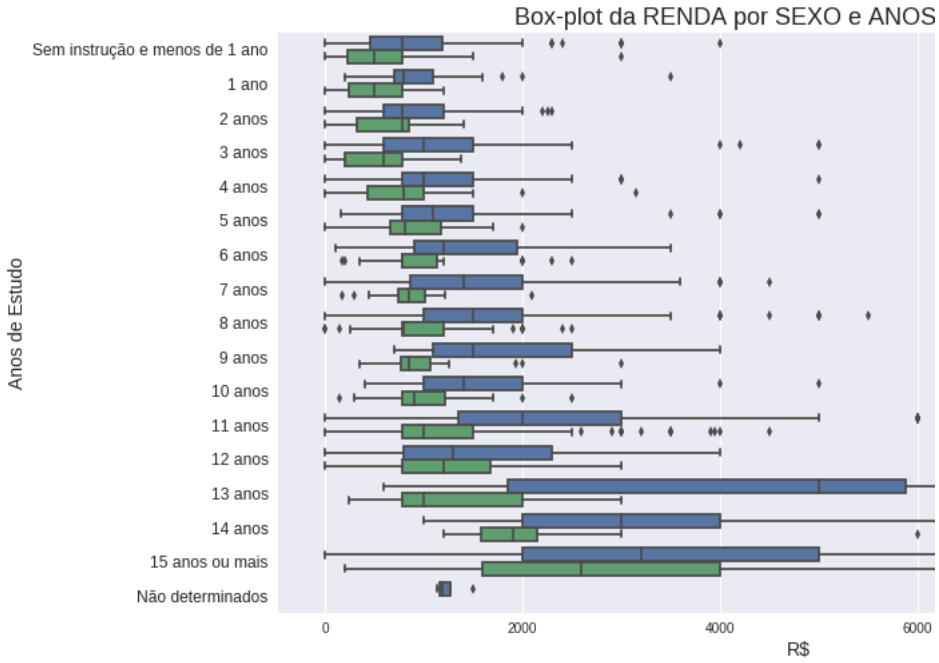
	max		mean		median		std	
Sexo	Masculino	Feminino	Masculino	Feminino	Masculino	Feminino	Masculino	Feminino
Anos de Estudo								
Sem instrução e menos de 1 ano	30000	10000	799.49	516.20	700	390	1023.90	639.31
1 ano	30000	2000	895.63	492.77	788	400	1331.95	425.29
2 anos	40000	4000	931.18	529.91	788	450	1435.17	498.23
3 anos	80000	3500	1109.20	546.85	800	500	2143.80	424.12
4 anos	50000	10000	1302.33	704.28	1000	788	1419.82	629.55
5 anos	35000	8000	1338.65	781.39	1045	788	1484.65	635.78
6 anos	25000	6000	1448.88	833.73	1200	788	1476.63	574.55
7 anos	40000	9000	1465.50	830.75	1200	788	1419.71	602.04
8 anos	30000	18000	1639.40	933.62	1300	800	1515.58	896.78
9 anos	60000	20000	1508.04	868.02	1200	788	2137.66	973.22
10 anos	45000	6000	1731.27	925.92	1218	800	2078.61	620.61
11 anos	200000	100000	2117.06	1286.79	1500	1000	2676.54	1819.04
12 anos	30000	120000	2470.33	1682.31	1800	1200	2268.08	4851.83
13 anos	25000	20000	3195.10	1911.73	2400	1300	2797.12	2053.79
14 anos	50000	20000	3706.62	2226.46	2500	1600	3987.21	2064.08
15 anos ou mais	200000	100000	6134.28	3899.51	4000	2800	7447.61	4212.77
Não determinados	7000	3000	1295.76	798.17	1200	788	979.65	459.99

Com esta tabela, podemos treinar nossas capacidades analíticas e analisar as características mais importantes.

Depois, construímos o `boxplot` da variável `Renda` de acordo com os `Anos de Estudo` e `Sexo`.

Em `set_yticklabels()`, passamos uma lista com os valores dos `anos_de_estudo` com `values()` por meio de `for` para os valores com os nome

```
ax = sns.boxplot(x = 'Renda', y = 'Anos de Estudo', hue = 'Sexo', data=dados.query('Renda < 10000 and Idade

ax.figure.set_size_inches(14, 8)    # Personalizando o tamanho da figura

ax.set_title('Box-plot da RENDA por SEXO e ANOS DE ESTUDO', fontsize=18)    # Configurando o título do grá

ax.set_xlabel('R$', fontsize=14)    # Configurando o label do eixo X

ax.set_ylabel('Anos de Estudo', fontsize=14)    # Configurando o label do eixo Y
ax.set_yticklabels([key for key in anos_de_estudo.values()], fontsize=12)    # Configurando o label de cad

# Configurações da legenda do gráfico (Sexo)
handles, _ = ax.get_legend_handles_labels()
ax.legend(handles, ['Masculino', 'Feminino'], fontsize=12)

ax
```



Notaremos várias configurações parecidas com o `boxplot` anterior.

Se quiséssemos somente as chaves do dicionário, aplicaríamos o método `key()` .

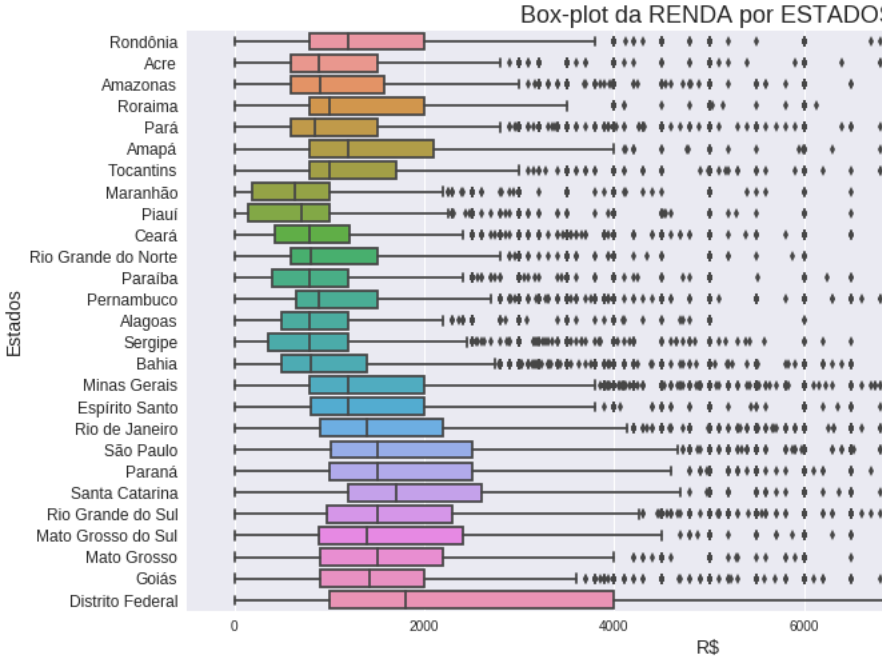
Depois, obteremos a média, mediana, valor máximo e desvio padrão da variável `Renda` agrupadas por `UF` .

Nesta resposta, utilizamos o `groupby()` e o `agg()` conforme as dicas escritas no notebook.

```
renda_estatisticas_por_uf = dados.groupby(['UF']).agg({'Renda': ['mean', 'median', 'max', 'std']})
renda_estatisticas_por_uf.rename(index = uf)
```

Por fim, construímos um `boxplot()` desses dados com uma seleção de até R$10.000,00 de rendimento mensal.

```
ax = sns.boxplot(x = 'Renda', y = 'UF', data=dados.query('Renda < 10000'), orient='h')

ax.figure.set_size_inches(14, 8)    # Personalizando o tamanho da figura

ax.set_title('Box-plot da RENDA por ESTADOS', fontsize=18)    # Configurando o título do gráfico

ax.set_xlabel('R$', fontsize=14)    # Configurando o label do eixo X

ax.set_ylabel('Estados', fontsize=14)    # Configurando o label do eixo Y
ax.set_yticklabels([key for key in uf.values()], fontsize=12)    # Configurando o label de cada categoria

ax
```



Com este gráfico, poderemos tirar diversas conclusões sobre as características de distribuição da variável.

É possível sempre fazermos diversas análises com esses conhecimentos estatísticos, inclusive com outros datasets.